WRDC-TR-90-8007
Volume VIII
Part 19

INTEGRATED INFORMATION SUPPORT SYSTEM (IISS)
Volume VIII - User Interface Subsystem
Part 19 - Forms Driven Editor Development Specification

S. Barker

Control Data Corporation
Integration Technology Services
2970 Presidential Drive
Fairborn, OH 45324-6209

September 1990

Final Report for Period 1 April 1987 - 31 December 1990

MANUFACTURING TECHNOLOGY DIRECTORATE
WRIGHT RESEARCH AND DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6533

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever, regardless whether or not the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data. It should not, therefore, be construed or implied by any person, persons, or organization that the Government is licensing or conveying any rights or permission to manufacture, use, or market any patented invention that may in any way be related thereto.

This technical report has been reviewed and is approved for publication.

This report is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations

DAVID L. JUDSON, Project Manager
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

FOR THE COMMANDER:

BRUCE A. RASMUSSEN, Chief
WRDC/MTI
Wright-Patterson AFB, OH 45433-6533

25 July 91
DATE

If your address has changed, if you wish to be removed form our mailing list, or if the addressee is no longer employed by your organization please notify WRDC/MTI, Wright-Patterson Air Force Base, OH 45433-6533 to help us maintain a current mailing list.

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release; Distribution is Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) DS 620344402	5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-90-8007 Vol. VIII, Part 19
6a. NAME OF PERFORMING ORGANIZATION Control Data Corporation; Integration Technology Services 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION WRDC/MTI
6c. ADDRESS *(City,State, and ZIP Code)* 2970 Presidential Drive Fairborn, OH 45324-6209	7b. ADDRESS *(City, State, and ZIP Code)* WPAFB, OH 45433-6533
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Wright Research and Development Center, Air Force Systems Command, USAF 8b. OFFICE SYMBOL *(If applicable)* WRDC/MTI	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUM. F33600-87-C-0464
8c. ADDRESS *(City, State, and ZIP Code)* Wright-Patterson AFB, Ohio 45433-6533	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
78011F	595600	F95600	20950607

11. TITLE Forms I	See block 19
12. PERSONAL AUTHOR(S) Structural Dynamics Research Corporation: Barker, S., Glandorf, F., Jones, L.	

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr.,Mo.,Day)*	15. PAGE COUNT
Final Report	4/1/87-12/31/90	1990 September 30	55

16. SUPPLEMENTARY NOTATION

WRDC/MTI Project Priority 6203

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify block no.)*
FIELD	GROUP	SUB GR.	
1308	0905		

19. ABSTRACT *(Continue on reverse if necessary and identify block number)*

This specification establishes the design, development, test, and qualification requirements of the Forms Driven Form Editor (FDFE).

BLOCK 11:

INTEGRATED INFORMATION SUPPORT SYSTEM
Vol VIII - User Interface Subsystem

Part 19 - Forms Driven Editor Development Specification

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT	21. ABSTRACT SECURITY CLASSIFICATION	
UNCLASSIFIED/UNLIMITED x SAME AS RPT. DTIC USERS	Unclassified	
22a. NAME OF RESPONSIBLE INDIVIDUAL David L. Judson	22b. TELEPHONE NO. *(Include Area Code)* (513) 255-7371	22c. OFFICE SYMBOL WRDC/MTI

EDITION OF 1 JAN 73 IS OBSOLETE

DD FORM 1473, 83 APR

FOREWORD

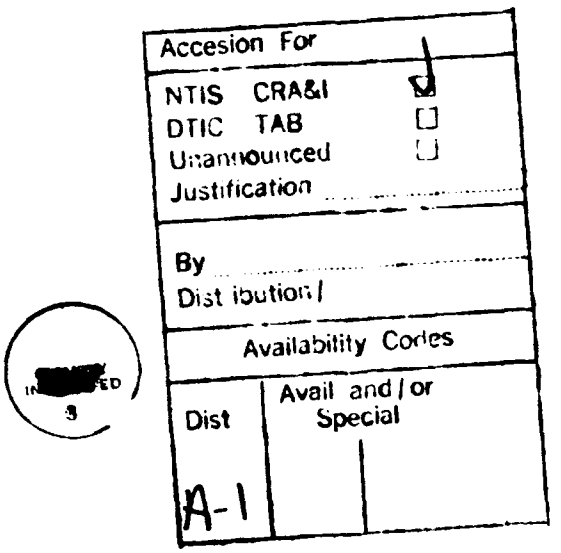
This technical report covers work performed under Air Force Contract F33600-87-C-0464, DAPro Project. This contract is sponsored by the Manufacturing Technology Directorate, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio. It was administered under the technical direction of Mr. Bruce A. Rasmussen, Branch Chief, Integration Technology Division, Manufacturing Technology Directorate, through Mr. David L. Judson, Project Manager. The Prime Contractor was Integration Technology Services, Software Programs Division, of the Control Data Corporation, Dayton, Ohio, under the direction of Mr. W. A. Osborne. The DAPro Project Manager for Control Data Corporation was Mr. Jimmy P. Maxwell.

The DAPro project was created to continue the development, test, and demonstration of the Integrated Information Support System (IISS). The IISS technology work comprises enhancements to IISS software and the establishment and operation of IISS test bed hardware and communications for developers and users.

The following list names the Control Data Corporation subcontractors and their contributing activities:

SUBCONTRACTOR	ROLE
Control Data Corporation	Responsible for the overall Common Data Model design development and implementation, IISS integration and test, and technology transfer of IISS.
D. Appleton Company	Responsible for providing software information services for the Common Data Model and IDEF1X integration methodology.
ONTEK	Responsible for defining and testing a representative integrated system base in Artificial Intelligence techniques to establish fitness for use.
Simpact Corporation	Responsible for Communication development.
Structural Dynamics Research Corporation	Responsible for User Interfaces, Virtual Terminal Interface,and Network Transaction Manager design, development, implementation, and support.
Arizona State University	Responsible for test bed operations and support.

TABLE OF CONTENTS



LIST OF ILLUSTRATIONS

SECTION 1

SCOPE

1.1 Identification

This specification establishes the development, test and qualification requirements of a computer program identified as the Forms Driven Form Editor (FDFE) and the design of the FDFE. The FDFE is one configuration item of the Integrated Information Support System (IISS) User Interface.

1.2 Functional Summary

The Forms Driven Form Editor is a software tool for creating and initializing form definitions. The FDFE displays a series of screens which request information from the user and visually show the form under construction. Once a form has been completed, the FDFE stores the form definition constructs needed to recreate the form. The stored form can be selected and modified.

The FDFE functions can be viewed on three different levels:

A. Conceptual

The user views the FDFE as a tool which can perform the following functions on form language source:

1) insert, select, modify, and drop form language source
2) insert, select, modify, and drop forms within a forms language source
3) select form language source and combine sections of one or more form language sources into one form language source
4) list all the form language sources
5) rename and copy form language source
6) list all forms created in a form language source

The FDFE may be used by the editor to perform the following operations on the compiled form definitions:

1) drop the compiled form definition
2) list all the compiled form definitions
3) view the compiled form definition

The fields of a form may be operated on in the following ways:

1) insert, drop, modify, select and list fields
2) copy fields from one form to another form
3) for these operations, all the different edit modes may be used

B. External

The runtime UI or UIMS views the FDFE, which is part of the UIDS, as an application program which uses the form processor. Data to be selected or stored comes from or is passed to the Common Data Model (CDM) in the integrated implementation; otherwise, a file system is used. The FDFE also interacts with the Forms Language Compiler (FLAN) to translate between the forms language source and the compiled form definition.

C. Internal

The FDFE is a C program which makes extensive use of form language sources and compiled forms, performs interactive user input/output via the Form Processor (FP), and uses the FP to manage the compiled forms. The internal form data structure is the same as that used by the Form Processor.

Since editors require extensive testing and revision before achieving true functionality, changes to the presented design may occur through use of the FDFE to design and develop forms for application.

SECTION 2

DOCUMENTS

2.1 Reference Documents

[1] General Electric Co., ICAM Integrated Support System (IISS) Test Bed System Design Specification (Draft), 7 Feb 83, SDS620140000.

[2] Systran, ICAM Documentation Standards, 15 September 1983, IDS150120000C.

[3] Structural Dynamics Research Corporation, Form Processor User Manual, UM 620244200A, 16 Feburary 1987.

[4] Structural Dynamics Research Corporation, Report Writer Development Specification, DS 620244501A, 16 Feburary 1987.

[5] Structural Dynamics Research Corporation, Rapid Application Generator Development Specification DS 620244502A, 16 Feburary 1987.

[6] Structural Dynamics Research Corporation, Text Editor Development Specification, DS 620244600A, 16 February 1987.

[7] Structural Dynamics Research Corporation, Form Processor Development Specification, DS 620244200, 16 February 1987.

[8] Structural Dynamics Research Corporation, Application Interface Development Specification, DS 620244700A, 16 February 1987.

[9] Structural Dynamics Research Corporation, Forms Language Compiler Development Specification, DS620244401B, 8 December 1987.

[10] Structural Dynamics Research Corporation, Forms Driven Form Editor Development Specification, DS 620244402A, 16 February 1987.

[11] Structural Dynamics Research Corporation, User Interface Services Development Specification, DS 620244100A, 16 February 1987.

[12] Structural Dynamics Research Corporation, Virtual Terminal Development Specification, DS 620244300A, 16 February 1987.

[13] General Electric Corporation, IISS System Design Specification, SDS 620140000, 7 February 1983.

[14] Structural Dynamics Research Corporation, IISS Form Processor Application Interface, DS 620244700A, 16 February 1987.

[15] Structural Dynamics Research Corporation, Form Editor User's Manual, UM 620244400A, 16 February 1987.

[16] Structural Dynamics Research Corporation, Rapid Application Generator User's Manual, UM 620244502A, 16 February 1987.

[17] American National Standards Institute, Programmer's Hierarchical Interactive Graphics System, dpANS X3.144 (Draft Proposed Standard X3H3/87-100).

[18] American National Standards Institute, Graphical Kernel System Functional Description, ANSI X3.124-1985.

[19] International Organization for Standardization, Graphical Kernel System (GKS) functional description, ISO 7942-1985.

[20] American National Standards Institute, Computer Graphics Metafile for the Storage and Transfer of Picture Description Information, ANSI X3.122-1986.

[21] Structural Dynamics Research Corporation, C Coding Guidelines, IISS Programmer's Guide

[22] American National Standards Institute, Additional Controls for use with American National Standard Code for Information Interchange, ANSI X3.64-1975.

2.2 Terms and Abbreviations

American Standard Code for Information Interchange: (ASCII), the character set defined by ANSI X3.4 and used by most computer vendors.

Application Generator (AG): A subset of the IISS User Interface that consists of software modules that generate IISS application code and associated form definitions based on a language input. The part of the AG that generates report programs is called the Report Writer. The part of the AG that generates interactive applications is called the Rapid Application Generator.

Application Interface: (AI), subset of the IISS User Interface that consists of the callable routines that are linked with applications that use the Form Processor or Virtual Terminal. The AI enables applications to be hosted on computers other than the host of the User Interface.

Application Process: (AP), a cohesive unit of software that can be initiated as a unit to perform some function or functions.

Archive: A file containing the definitions of one or more structures. Structures can be saved into or retrieved from an archive by means of subroutine calls.

Attribute: field characteristic such as blinking, highlighted, black, etc. and various other combinations. Background attributes are defined for forms or windows only. Foreground attributes are defined for items. Attributes may be permanent, i.e., they remain the same unless changed by the application program, or they may be temporary, i.e., they remain in effect until the window is redisplayed.

Cell Array: A geometric primitive which consists of a number of adjoining colored or shaded parellelograms.

Closed Figure: A figure is closed if the path traced by a moving point returns to its starting position. The starting position may be arbitrarily assigned. "Fillarea" is synonymous with "closed figure".

Common Data Model: (CDM), IISS subsystem that describes common data application process formats, form definitions, etc. of the IISS and includes conceptual schema, external schemas, internal schemas, and schema transformation operators.

Complex Figure: A figure is complex if the path traced by a moving point crosses itself. An arbitrary point may be determined to be contained within the traced boundary if a line drawn to infinity crosses the boundary an odd number of times. If the number of crossings is zero or even, the point is outside the traced boundary.

Computer Graphics Metafile (CGM): A file with a standardized format which is used to store or transmit graphic images.

Computer Program Configuration Item: (CPCI), an aggregation of computer programs or any of their discrete portions, which satisfies an end-use function.

Conceptual Schema: (CS), the standard definition used for all data in the CDM. It is based on IDEF1 information modelling.

Current Cursor Position: the position of the cursor before an edit command or function is issued in the text editor.

Cursor Position: the position of the cursor after any command is issued.

Dependent Data: Data correlated to a dependent variable.

Dependent Variable: A mathematical variable whose value is determined by that of one or more other variables in a function.

Device Drivers: (DD), software modules written to handle I/O for a specific kind of terminal. The modules map terminal specific commands and data to a neutral format. Device Drivers are part of the UI Virtual Terminal.

Display List: An internal Form Processor list that contains only those forms that have been added to the screen and are currently displayed on the screen, along with information on where those forms are used.

Display Size: the number of lines used in the edit area.

Element: A graphics line or other primitive composed of graphics lines, such as an arc.

Extended Binary Coded Decimal Interchange Code: (EBCDIC), the character set used by a few computer vendors (notably IBM) instead of ASCII.

External Schema: (ES), an application's view of the CDM's conceptual schema.

Field: In reference to the Forms Processor, "field" refers to any object on the open or display list. These objects can be forms, items, window, etc. In reference to graphs, "field" refers to a collection of one or more graph figures. A graph field can be an axis, curve, pie chart, grid, etc.

Field Pointer: indicates the ITEM which contains the current cursor position.

Figure: A collection of elements. A figure may be closed or open.

Fill Area: A geometric primitive consisting of a planar area which is to be filled in with a particular color or pattern.

Form: structured view which may be imposed on windows or other forms. A form is composed of fields. These fields may be defined as forms, items, and windows, prompts, non-graphics lines, and structures.

Form Definition: (FD), form definition language after compilation. It is read at runtime by the Form Processor.

Form Definition Language: (FDL), the language in which electronic forms are defined.

Forms Driven Form Editor: (FDFE), subset of the Form Editor which consists of a forms-driven application used to create Form Definition files interactively.

Form Editor: (FE), subset of the IISS User Interface that is used to create definitions of forms. The FE consists of the Forms Driven Form Editor (FDFE) and the Forms Language Compiler (FLAN).

Form Hierarchy: a graphic representation of the way in which forms, items and windows are related to their parent form.

Form Language Compiler: (FLAN), subset of the Form Editor that consists of a batch process that accepts a series of form definition language (FDL) statements and produces form definition files as output.

Form Processor: (FP), subset of the IISS User Interface that consists of a set of callable execution time routines available to an application program for form processing.

Form Processor Text Editor: (FPTE), subset of the Form Processor that consists of software modules that provide text editing capabilities to all users of applications that use the Form Processor.

Generalized Drawing Primitive (GDP): A geometric primitive whose exact definition and representation is not specified. This allows an implementation to support additional geometric primitives such as arcs or conic sections in a standard conforming manner.

Graph: A picture correlated with data that alters as the data changes; by necessity, this is a dynamic (not predefined) picture. A graph may be imposed on windows or forms.

Graph Definition Language (GDL): An extension of the Forms Definition Language (FDL) which is used to define business graphs such as pie charts, X-Y plots, and bar charts.

Graph Figure: A collection of graphics primitives. The primitives can be circles, lines, arcs, etc.

Graphical Kernel System (GKS): A 2-dimensional graphics standard which is defined independently of any programming language.

Icon: A collection of figures and points that is predefined. An icon may be imposed on windows and forms. "Icon" is synonymous with "picture".

IISS Function Screen: the first screen that is displayed after logon. It allows the user to specify the function he wants to access and the device type and device name on which he is working.

Independent Data: Data that is correlated to an independent variable.

Independent Variable: A mathematical variable whose value is specified first and determines the value of one or more other values in an expression or function. For example, in a business graph of sales versus month, month is the independent variable and sales is the dependent variable, because sales varies by month.

Integrated Information Support System: (IISS), a test computing environment used to investigate, demonstrate and test the concepts of information management and information integration in the context of Aerospace Manufacturing. The IISS addresses the problems of integration of data resident on heterogeneous data bases supported by heterogeneous computers interconnected via a Local Area Network. (LAN).

Item: non-decomposable area of a form in which hard-coded descriptive text may be placed and the only defined areas where user data may be input/output.

Local Area Network (LAN): A privately owned network that offers reliable, high-speed communications channels optimized for connecting information processing equipment in a limited geographic area.

Logical Device: a conceptual device that identifies a top level window of an application. It is used to distinguish between multiple applications running simultaneously on a physical device. NOTE that a single application can have more than one logical device. To the end user, this also appears as multiple applications running simultaneously.

Message: descriptive text which may be returned in the standard message line on the terminal screen. They are used to warn of errors or provide other user information.

Message Line: a line on the terminal screen that is used to display messages.

Network Transaction Manager: (NTM), IISS subsystem that performs the coordination, communication and

housekeeping functions required to integrate the Application Processes and System Services resident on the various hosts into a cohesive system.

Open Figure: A figure is open if the path traced by a moving point does not return to its starting position. The starting position may be arbitrarily assigned. "Polyline" is synonymous with "open figure".

Open List: An internal Form Processor list that contains all forms that the application has opened for use along with information on where the form is used.

Operating System: (OS), software supplied with a computer which allows it to supervise its own operations and manage access to hardware facilities such as memory and peripherals.

Page: An instance of form in a window that is created whenever a form is added to a window.

Paging and Scrolling: a method which allows a form to contain more data than can be displayed with provisions for viewing any portion of the data buffer.

Physical Device: a hardware terminal.

Picture: A collection of figures and points that is predefined. A picture may be imposed on a window or a form. "Picture" is synonymous with "icon".

Polyline: A geometric primitive consisting of one or more connected line segments.

Polymarker: A geometric primitive consisting of one or more marker symbols (such as a cross or a dot).

Presentation Schema: (PS), may be equivalent to a form. It is the view presented to the user of the application.

Previous Cursor Position: the position of the cursor when the previous edit command was issued.

Programmer's Hierarchical Interactive Graphics System (PHIGS): A two and three dimensional graphics draft standard which is defined independently of any programming language.

Qualified Name: the name of a form, item or window preceded by the hierarchy path so that it is uniquely identified.

Report Definition Language: an extension of the Forms Definition Language that includes retrieval and calculation of database information and is used to define reports.

Structure: A collection of graphic primitives much as a form is a collection of textual primitives.

Subform: a form that is used within another form.

Text: A geometric primitive consisting of a number of characters with a particular orientation arranged along a particular path and aligned in a particular manner with some point. In PHIGS, text may be specified as a part of the image that participates fully in all transformations or as annotation which remains in the plane of the screen at all times.

Text Editor (TE): A subset of the IISS User Interface that consists of a file editor that is based on the test editing functions built into the Form Processor (FP).

User Data: Data which is either input by the user or output by the application programs to items.

User Interface: (UI), A subsystem of IISS that controls the user's terminal and interfaces with the rest of the system. The UI consists of two major subsystems: the User Interface Development System (UIDS) and the User Interface Management System (UIMS).

User Interface Development System: (UIDS), collection of IISS User Interface subsystems that are used by applications programmers as they develop IISS applications. The UIDS includes the Form Editor (FE) and the Application Generator (AG).

User Interface Management System: (UIMS), the runtime UI. It consists of the Form Processor (FP), Virtual Terminal (VT), Application Interface (AI), the User Interface Services (UIS) and the Text Editor (TE).

User Interface Monitor: (UIM), part of the Form Processor that handles messaging between the NTM and the UI. It also provides authorization checks and initiates applications.

User Interface Services (UIS): A subset of the IISS User Interface that consists of a package of routines that aid users in controlling their environment. It inclused message management, change password, and application definition services.

User Interface/Virtual Terminal Interface: (UI/VTI), another name for the User Interface.

Virtual Terminal: (VT), subset of the IISS User Interface that performs the interfacing between different terminals and the UI. This is done by defining a specific set of terminal features and protocols which must be supported by the UI software which constitutes the virtual terminal definition. Specific terminals

are then mapped against the virtual terminal software by specific software modules written for each type of real terminal supported.

Virtual Terminal Interface (VTI): The protocol used to communicate with a device driver.

Window: dynamic area of a terminal screen on which predefined forms may be placed at run time.

Window Manager: a facility which allows the following to be manipulated: size and location of windows, the device on which an application is running, the position of a form within a window. It is part of the Form Processor(FP).

Workstation: an abstract graphical workstation which provides the logical interface to the applications program. It is analogous to a form in a window.

SECTION 3

REQUIREMENTS

3.1 Computer Program Definition

The Forms Driven Form Editor provides an interactive tool for constructing and viewing forms. The tool provides the capability to store and select user forms.

3.1.1 Interface Requirements

The Forms Driven Form Editor (FDFE) interfaces directly with users as a UIS application. Physical terminals are assumed to have a video display, a textual keyboard, four cursor positioning keys or key sequences, and 18 function keys (0 - 17) or key sequences. If the Icon Edit mode is to be used, then the terminal must support graphic output and input. The FDFE must interface with the AI, FLAN and the Operating System.

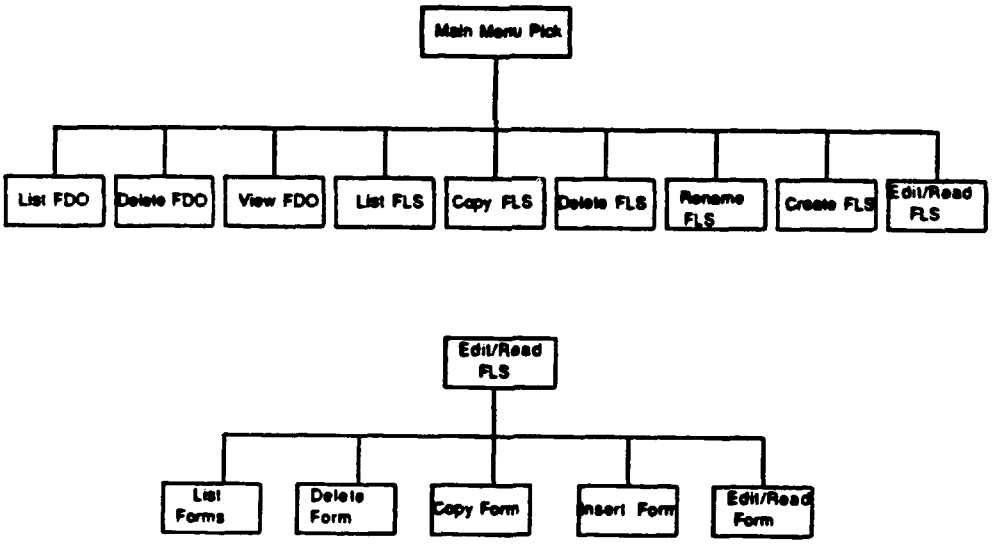


Figure 3-1 Interface Diagram

3.1.1.1 Application Interface

The FDFE interacts with users by calling appropriate routines of the Application Interface (AI). This interface creates messages which are sent to the Form Processor which moves information describing interactive terminal input and output and provides a link to users of the FDFE through the Virtual Terminal.

3.1.1.2 Forms Language Compiler Interface

The FDFE uses the Forms Language Complier (FLAN) to convert forms language source into the Form Processor internal forms structure. The FDFE also invokes the FLAN when the form under construction is to be stored.

3.1.1.3 Operating System Interface

The FDFE stores form language source files (fdl files on the VAX) and compiled form definitions (fd files on the VAX). Form language source files may subsequently be compiled and displayed. The storage of the fdl files and fd files is system dependent. The VAX implementation uses the logicals IISSSLIB (for fdl files) to store/retrieve the files in/from the appropriate directory.

3.1.1.4 User Interface Displays

The layout of the screens discussed in the following section may be found in Appendix A.

The user interface provides WORK TASKS for constructing and modifying form language sources and compiled form definitions:

1) List Form Language Sources - list all form language sources

2) Copy Form Language Source - copy from a form language source to a new form language source

3) Rename Form Language Source - Rename old form language source name to a new form language source name

4) Insert, Drop, Modify, Select Form Language Source - operations on the form language source

5) List Compiled Form Definitions - list all compiled form definitions

6) Display Compiled Form Definition - display the form as a user would view the form

7) Drop Compiled Form Definition - drop the compiled form definition

The user may choose any of these menu operations by marking the appropriate selection and filling in the necessary parameters for the operation. Also available to the user is command entry capability. The user types the appropriate command (the capital letters of the menu operation indicate the command) with its needed parameters in the order specified by the menu operation. Parameters are separated by blanks.

For each selected WORK TASK, the appropriate screen is displayed to complete the task or lead the user to the next task to be completed.

For the List tasks of the WORK TASK, the next screen displayed is the list of form language sources or compiled form definitions.

For the Copy, Rename and Drop tasks, the screen displayed next is the same screen with a success/error message displayed. For the Display task, the next screen displayed is the form itself with the appropriate instructional message on how to proceed.

For the Insert, Modify and Select tasks, the next screen displayed is the EDIT TASKS screen which provides the following operations:

1) List Forms - lists all forms created in the current form language source

2) Write Forms - saves the form language source

3) Write and Compile Forms - save both forms language source and compiled forms definition

4) Exit Compile forms - saves form language source and compiled form definitions for all forms created in the form language source and returns to work task screen

5) Insert, Modify and Select Form - operations that are performed on a form which then requires an edit mode selection

6) Drop Form - drop the specified form from the form language source

For the List Forms task, the next screen displayed lists all the forms created by the current form language source. For the Drop, Write Forms and Write and Compile Forms tasks, the next screen returned is the EDIT TASK Screen which contains the appropriate success/error messages.

For the Insert, Modify and Select Form tasks, the next screen displayed depends on the edit mode chosen:

1) Single Field - the edit operations done on the form are done on a per field basis

2) Form - the edit operations done on the form may be done on all the fields at once

3) Layout - for positioning and constructing fields of a form

4) Icon - for constructing icons made up of two dimensional graphic primitives

Each mode contains several activities which facilitate use of that particular mode. The user is provided with visual feedback describing the mode/task combination in use at any particular time. The Work Tasks, Edit Tasks and Edit modes are described in detail in later sections.

3.1.1.5 Data Entry

The FDFE uses four types of data entry:

1) positional selection
2) textual selection
3) by-example
4) completion

Positional selection requires the user to position the cursor in specific locations and mark the selected activity. Textual selection provides a menu of items and requires entry of an abbreviated code to select an item. By-example allows entry of information as it will appear in the completed form. Completion data entry requires entry of text to complete a displayed form.

On the WORK TASKS, EDIT TASKS, FIELD EDIT and FORM EDIT screens, all but the by-example method of data entry is used. The user must choose, however, on the WORK TASK and EDIT TASK screens whether to use positional selection (using the menu specified operations) or textual selection (entering the abbreviated command, the capital letters of the descriptive text of the menu specified operations) in the command entry line. The LAYOUT EDIT and ICON EDIT screens are the only screens which also allow data entry by example.

3.1.1.6 Function Keys

The <ENTER> key always implies continuing with the next logical step in the completion of the current task. The <HELP> function key is used as defined by the Form Processor. Whenever the <HELP> key is pressed with the cursor positioned at an item, the help message or help form for that item is displayed. The help forms or messages for further describing items have not been presented but follow a consistent form so that the user is given complete information about the item for which help has been requested.

The <QUIT> key takes the user back to the previous logical screen. If the user returns to the previous logical screen without completing entry of the data on the current screen, this data is lost.

The <PF12> key is used in Layout Edit mode to move fields around on a form. The <PF16> key is used in Layout Edit Mode for getting to Field Edit Mode and in Field Edit Mode for getting to Layout Edit Mode. In Layout Edit Mode, the user positions the cursor at the field on the screen that is to be further created and presses the <PF16> key. The resulting

screen is the Field Edit Mode screen which the user may then complete. The <PF13> and <PF14> keys are used to go forwards and backwards through fields on a form in Field Edit Mode.

In Icon Edit Mode all function keys (0 - 17) are used. For a detail description of their use see Appendix B.

3.2 Detailed Function Requirements

The general approach is to view the FDFE as a hierarchy of functions. The FDFE screens associated with the interactive functions are presented in Appendix A.

3.2.1 Function Hierarchy

The following hierarchy chart shows the organization of the FDFE:

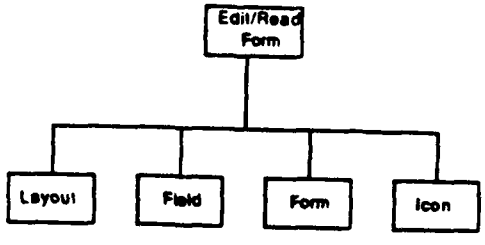


Figure 3-2 FDFE Hierarchy

3.2.2 Function Descriptions

The following paragraphs describe the functions associated with each of the major sections of the FDFE.

FUNCTION DEFINITIONS

FLS = Forms Language Source
FDO = Forms Definition Object

The main menu screen allows the user to choose among several file management options or to proceed to the edit task level.

3.2.2.1 List Existing FDO Files

This function lists all of the Forms Definition Object in the user's specified forms definition object library.

3.2.2.2 Delete an Existing FDO File

This function deletes a particular Forms Definition Object.

3.2.2.3 View an Existing Form (FDO File)

This function displays a form just as it would appear on the screen when used by a program.

3.2.2.4 List Existing FLS Files

This function lists all of the Forms Language Source in the user's specified forms language source library.

3.2.2.5 Copy an Existing FLS File

This function copies a particular Forms Language Source to another Forms Language Source of specified name.

3.2.2.6 Delete an Existing FLS File

This function deletes a particular Forms Language Source.

3.2.2.7 Rename an Existing FLS File

This function renames a particular Forms Language Source to a specified name

3.2.2.8 Create an FLS File

This function creates new internal data structure. It then presents the user with the form edit menu, from which the following functions can be chosen.

3.2.2.9 Edit / Read an Existing FLS File

This function retrieves a particular Forms Language Source and translates it into the internal data structure. It then presents the user with the form edit menu, from which the following functions can be chosen.

3.2.2.10 List Existing Forms

This function lists all forms in the Forms Language Source on which work is being done.

3.2.2.11 Delete an Existing Form

This function deletes a form from the Forms Language Source on which work is being done.

3.2.2.12 Copy an Existing Form

This function copies a Forms Language Source file into an alternate internal data structure and gets the pointer to the specified form.

3.2.2.13 Insert a Form

This function inserts a new form into the Forms Language Source on which work is being done and allows the user to enter information about the form.

3.2.2.14 Edit / Read an Existing Form

This function present user with current form information and if he / her is not in read only mode, allows him/her to update the information about the form.

3.2.2.15 Compile and Save FLS File Edited

This function saves a particular Forms Language Source (fdl file) and a Forms Definition Object (fd file) after it translates the internal data structure into forms language syntax.

3.2.3 Edit Modes

The following sections describe the modes available within the FDFE EDIT TASKS of Insert, Modify, and Select Form. Appendix A contains a layout of the associated screens that are used in each mode.

3.2.3.1 Field Edit Mode

The Field Edit Mode displays the form which describes the target form being constructed. The user enters textual information to complete information about the current form. The Field Edit Mode allows the user to individually drop, modify, select and insert fields within the current form. It collects the textual information entered by the user for purposes of modifying and retrieving a field. In addition, the user may select a field from a form other than the current form by completing the textual information of the name of the copy form language source and the copy form within the copy form language source. The user may navigate through all fields on the current form by using the select operation and specifying the field type and the direction, next or previous.

3.2.3.2 Layout Edit Mode

Layout Edit Mode provides "by-example" capabilities for defining forms. These capabilities can be divided into two categories. One category provides the ability to visually position subforms, items, and windows, to associate prompts with fields, and to visually move fields on the form. This capability saves form design time since positional parameters and field sizes are calculated for the user by Layout Edit Mode.

The second capability provides a convenient way to complete development of a form by using the Field Edit format to describe the necessary information about the fields positionally inserted in Layout Edit Mode. The user positions the cursor on the field to be further described and uses the <PF16> key to enter Field Edit Mode. This mode coaches the user on the necessary information to fully insert a field (see above).

The user is not required to go into the Field Edit Mode for the positionally inserted fields since the FDFE makes default assumptions about the required information for field definition. The following defaults are assumed: the field type is an item and the display attribute is INPUT. Field names are generated based on the row and column location of the field.

3.2.3.3 Form Edit Mode

The Form Edit Mode provides the user with a quick way of entering all the fields associated with a form at one time. The user may insert, modify, select, and drop multiple fields using this one form. The initial subform placed in the variable window allows the user to insert all the necessary information for any type of field: subform, window or item. The user may then further continue definition of item fields by marking the More Item Description operation contained within the subform. The user continues to do so until the item information is completed.

3.2.3.4 Icon Edit Mode

Icon Edit Mode provides "by-example" capabilities for defining forms which are to serve as icons. In this mode the user can draw an icon using the graphical primitive objects: ellipse, arc, box, and polyline. These objects can be moved, resized, rotated, and restacked to form the picture desired. The user may also choose the color and fill to be used for each of the primitive objects (for more detail see Appendix B).

3.2.4 Errors Management

The FDFE integrates with the Form Processor to handle three types of errors:

1) input syntax
2) input content
3) disruptive

Input syntax errors and errors in content values for items are handled within the module processing the user data coming in from its associated screens. Disruptive errors such as running out of memory or program bugs in Form Processor calls, can be caused by system activities and are handled by calling the standard system message routine which displays the message to the user on the current screen.

The FDFE modules use the Form Processor to provide proper error code mapping and, where possible, use existing FP error handling.

3.3 Performance Requirements

A performance goal is to achieve a one second response time following user input. This goal has been verified in several research efforts, but such a short time can rarely be achieved in large, centralized systems under realistic loads. Multi-processing systems which use microcomputers for terminals, with significant downloading, have been able to satisfy this goal.

A backup position for system performance is to provide a predictable response time for users. Users have been found to accept delays of several seconds if they occur in a repeatable manner. Users also will tolerate longer response times in situations where they perceive that the activity is complex.

3.3.1 Program Organization

The FDFE is organized such that access to data whether it be within the CDM or from a file system is only through calling the appropriate access routine. Also, not all modules are responsible for handling processing of the screens. In addition, a controlling model is responsible for forms handling for each level of the module hierarchy.

3.4 Human Performance

The following sections discuss several goals related to human performance considerations.

3.4.1 Minimization of Keypad Overloading

Each key should be used for only a small set of functions. Unambiguous input operations improve editor operation by reducing confusion. Positioning devices such as mice have been shown to be superior to keypads.

3.4.2 Overviewing

Complex forms may require representation on many screens. A method of presenting an abbreviated view of an entire parent form on one screen allows the user to navigate through the form by selecting subforms, windows, etc.

3.4.3 Graphical Representations and Icons

These approaches provide more information in limited screen landscapes and support a more powerful editor environment.

3.4.4 Undo Functions

Undo capability for previous actions facilitates exploration of editor functions for novices, and also allows experienced users to work at a more rapid pace. This feature was not implemented per se but mistakes may be corrected through careful use of the provided functions of the FDFE.

3.5 Data Base Requirements

3.5.1 Sources and Types of Input

The major source of input comes from processing of the user data entered on the FDFE forms. This input is error-checked and then translated into an internal structure. The internal structure is that used by the form processor. Another source of input is the form language source of the FDFE user.

3.5.2 Destinations and Types of Output

The output of the FDFE is the translation of the internal structure into the forms language source and the compiled form definition objects. This translation occurs at the point of saving the edited changes.

3.5.3 Internal Tables and Parameters

Internal tables consist of valid commands at the Work Task and Edit Task levels along with the number of required parameters for these commands.

3.6 Help Forms

Input items have an associated help message or help form which may be retrieved by pressing the <HELP> key.

SECTION 4

QUALITY ASSURANCE PROVISIONS

4.1 Introduction and Definition

"Testing" is a systematic process that may be preplanned and explicitly stated. Test techniques and procedures may be inserted in advance and a sequence of test steps may be specified. "Debugging" is the process of isolation and correction of the cause of an error.

4.2 Computer Programming Test and Evaluation

The quality assurance provisions for test consist of the normal testing techniques that are used during the construction process. They consist of design and code walk-throughs, unit testing, and integration testing. These tests are performed by the design team.

Each function is tested separately, then the entire sub-system is tested as a unit. All testing except for integration with software belonging to other companies is done at SDRC on the VAX.

The integration testing entails use of the scripting capability which executes major functions of the FDFE. The script file and its results may be used to verify the FDFE after release.

SECTION 5

PREPARATION FOR DELIVERY

The implementation site for the constructed software is the ICAM Integrated Support System (IISS) Test Bed site located at Arizona State University Tempe, Arizona. The software associated with each FDFE release is delivered on a medium which is compatible with the IISS Test Bed. The release is clearly identified and includes instructions on procedures to be followed for installation of the release.

APPENDIX A

FORMS DRIVEN FORM EDITOR SCREENS

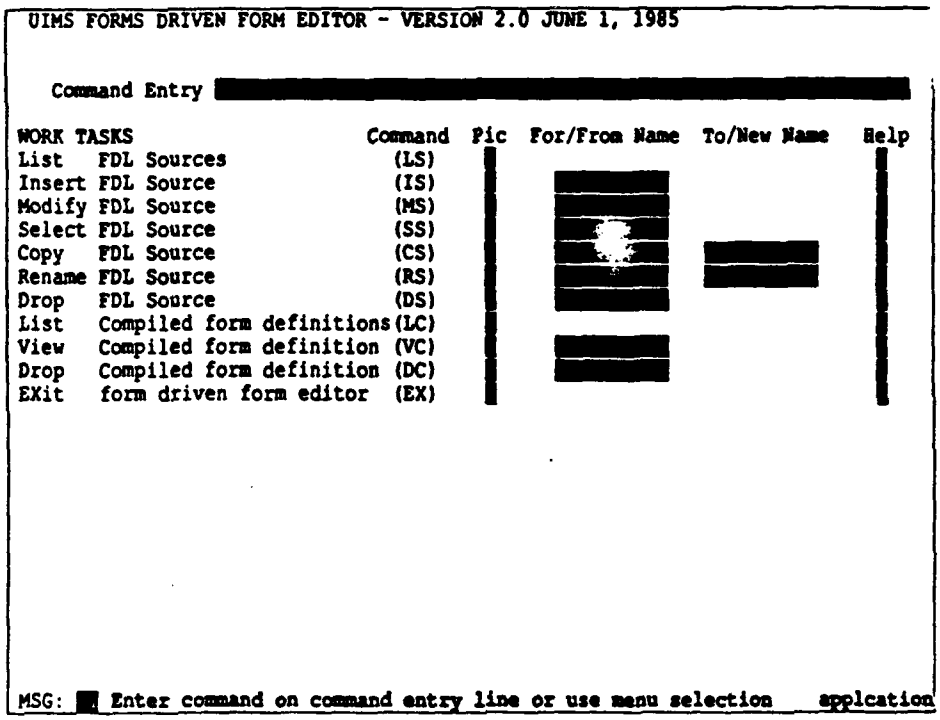


Figure A-1 Screen 1

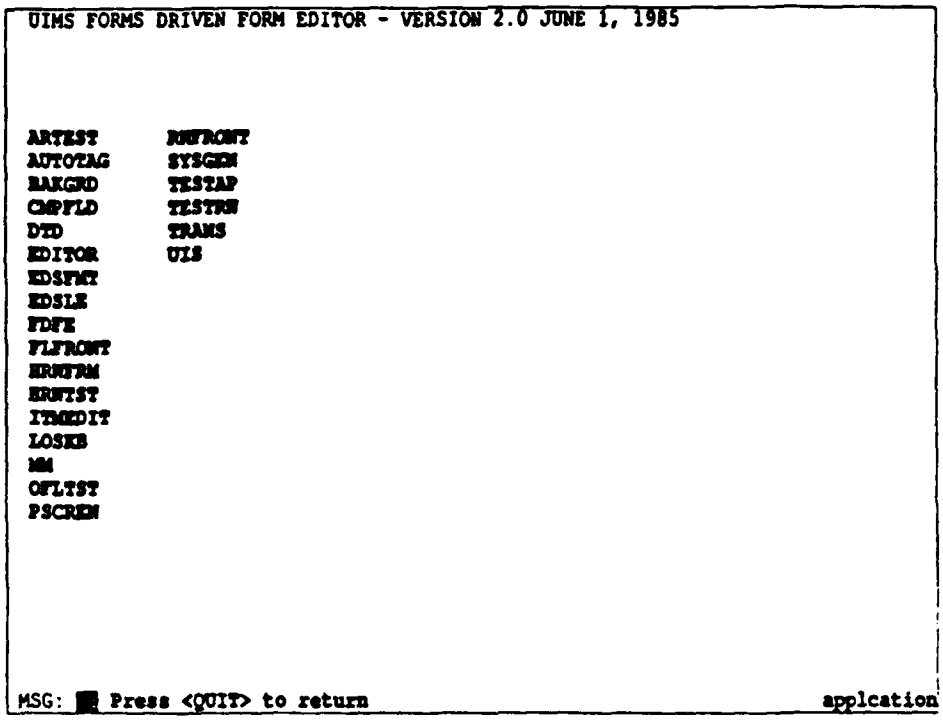


Figure A-2 Screen 2

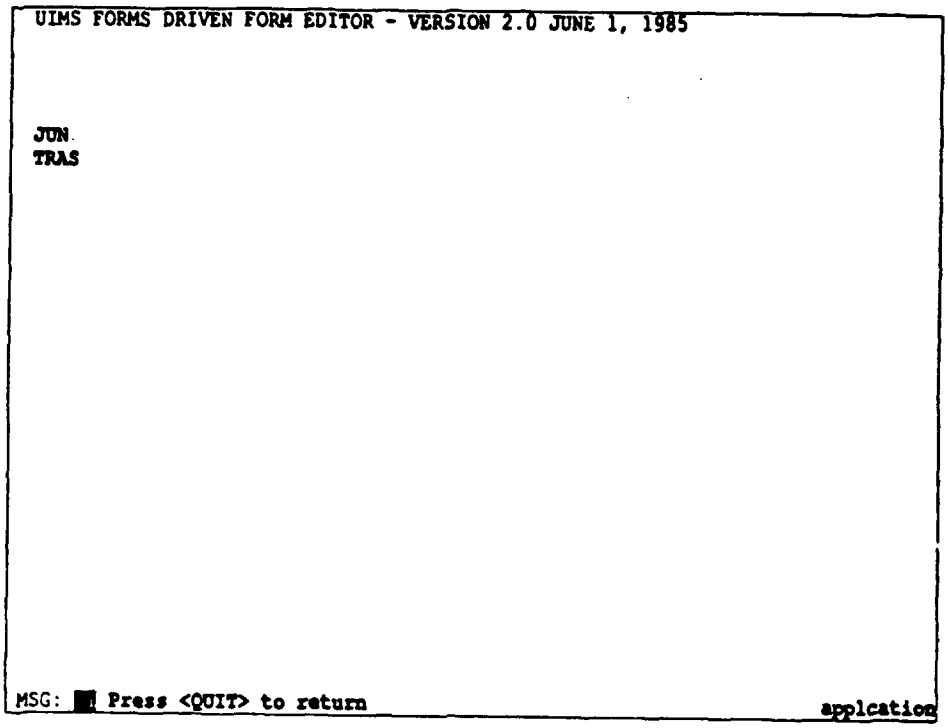


Figure A-3 Screen 3

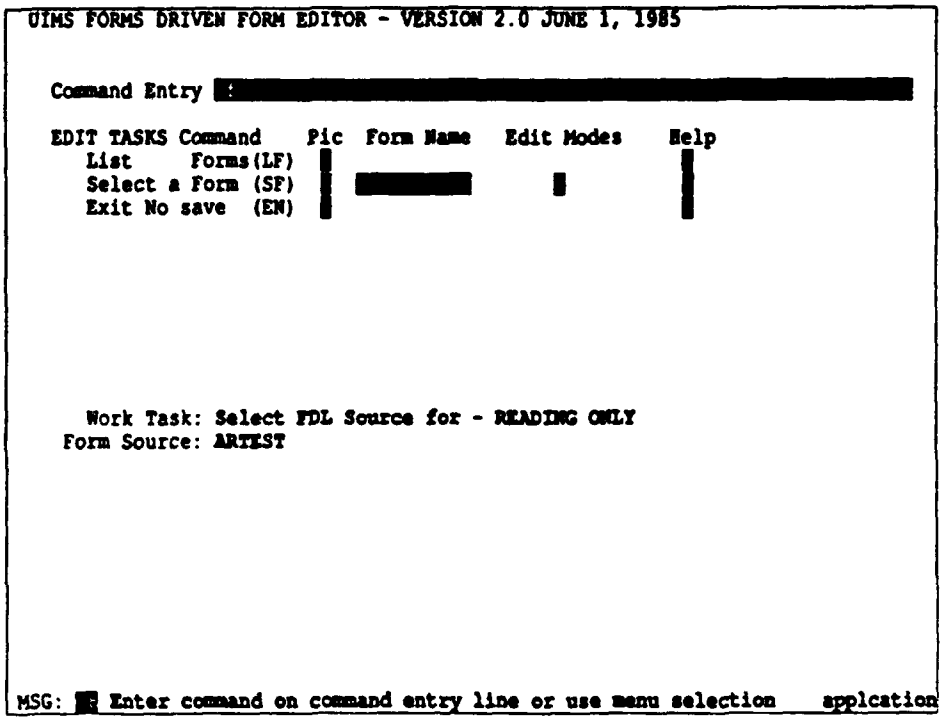


Figure A-4 Screen 4

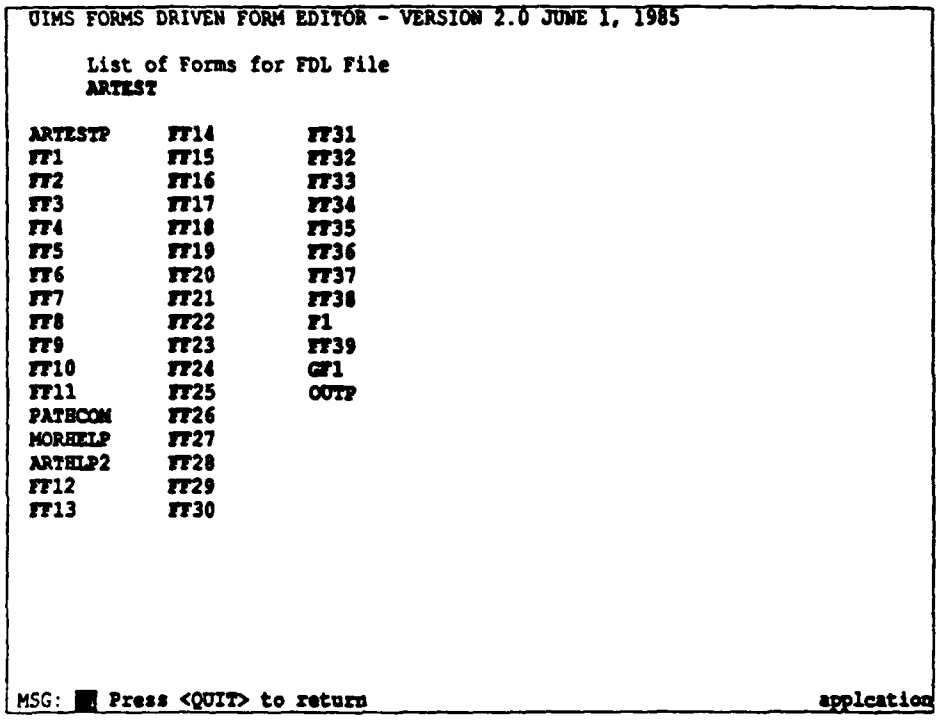


Figure A-5 Screen 5

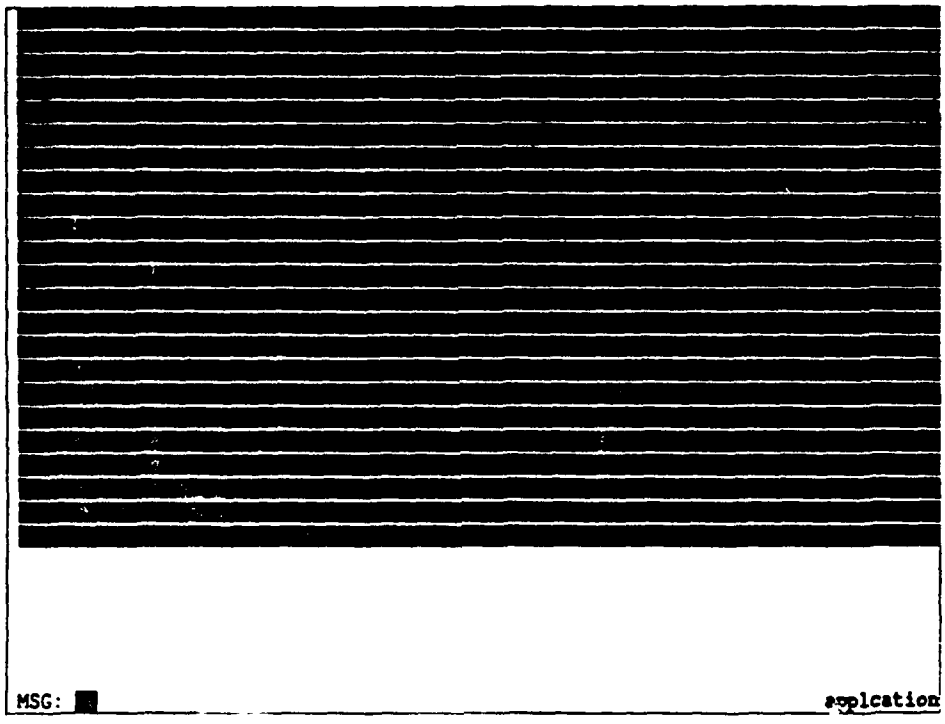


Figure A-6 Screen 6

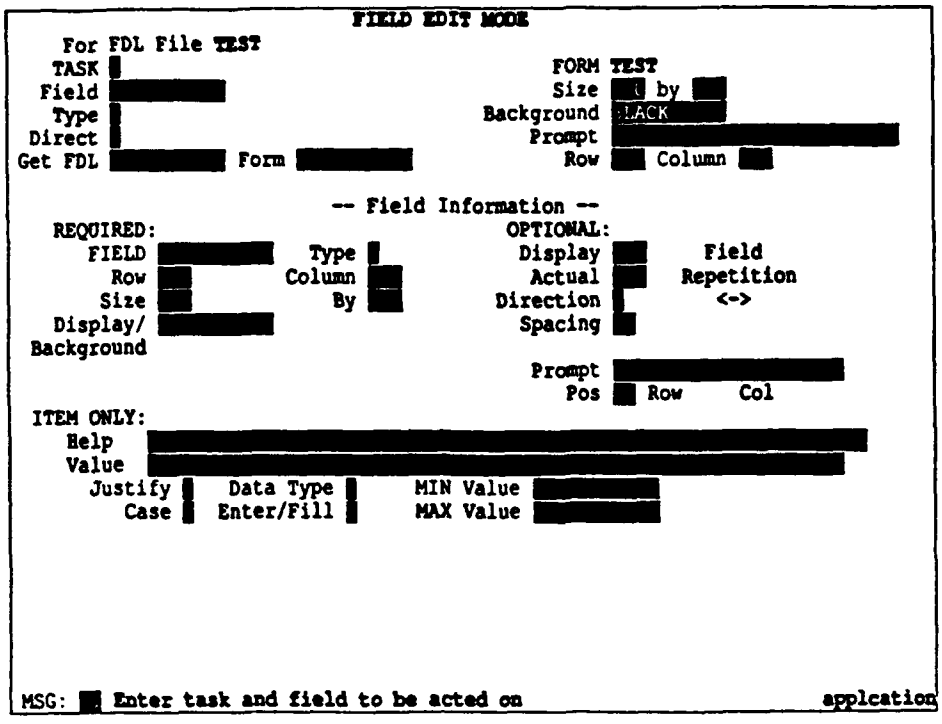


Figure A-7 Screen 7

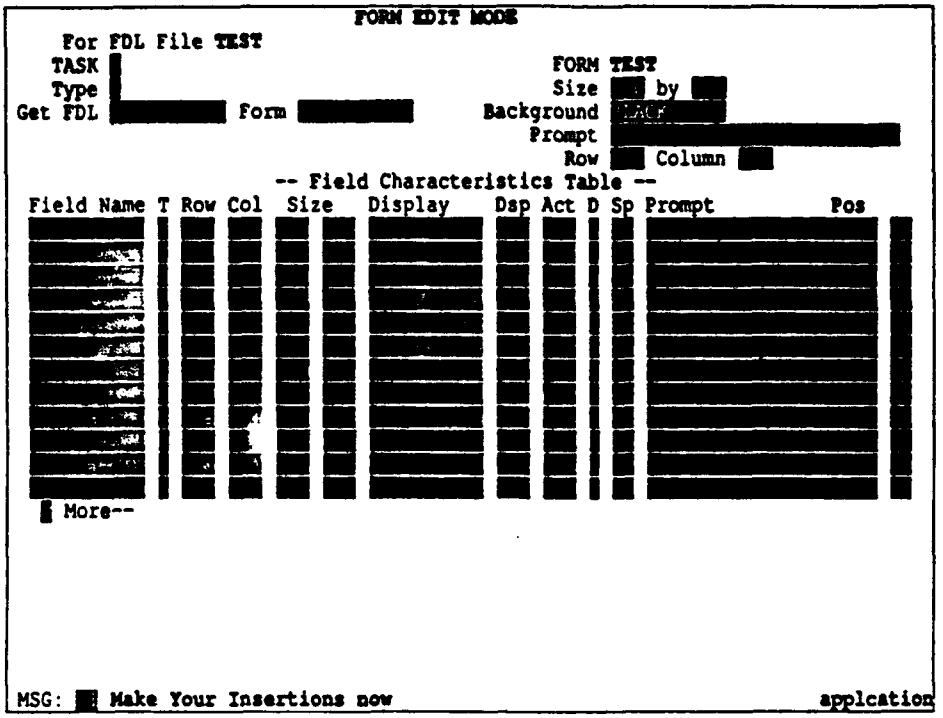


Figure A-8 Screen 8

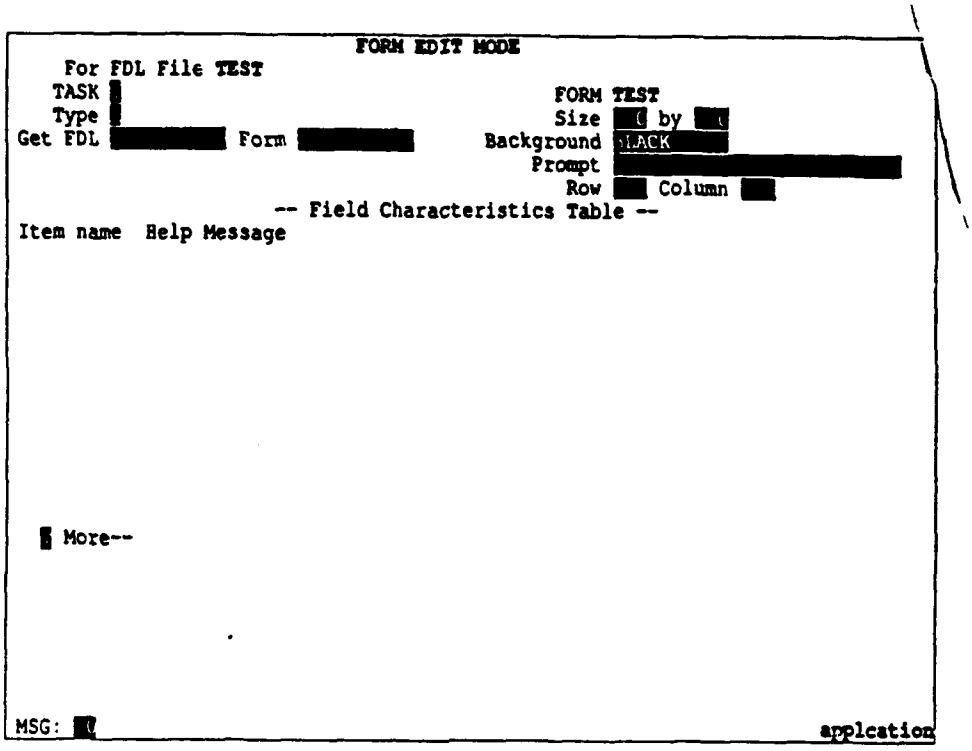


Figure A-9 Screen 9

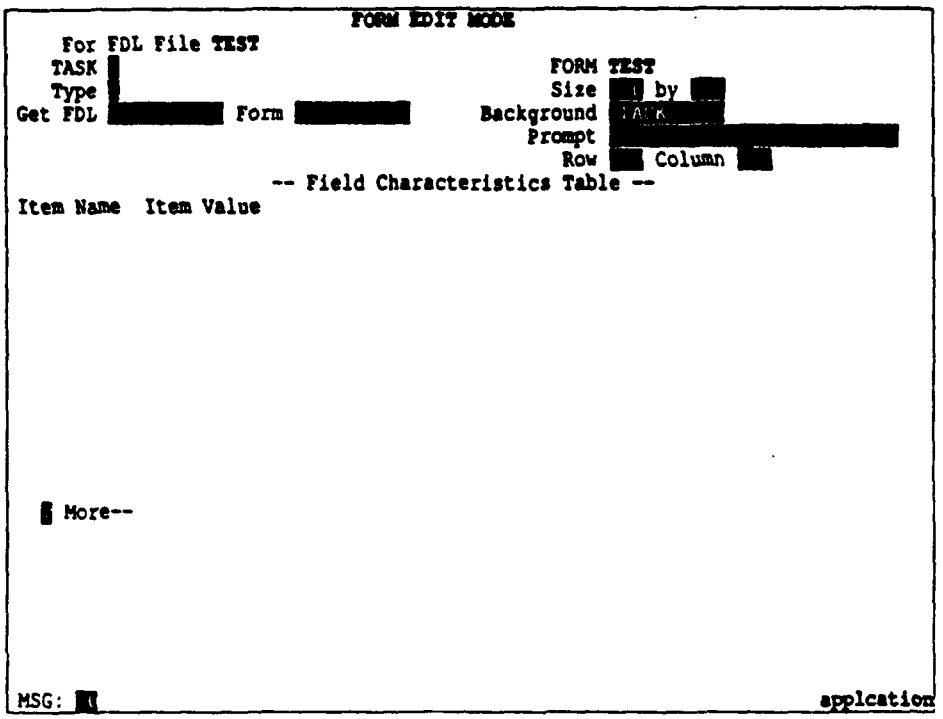


Figure A-10 Screen 10

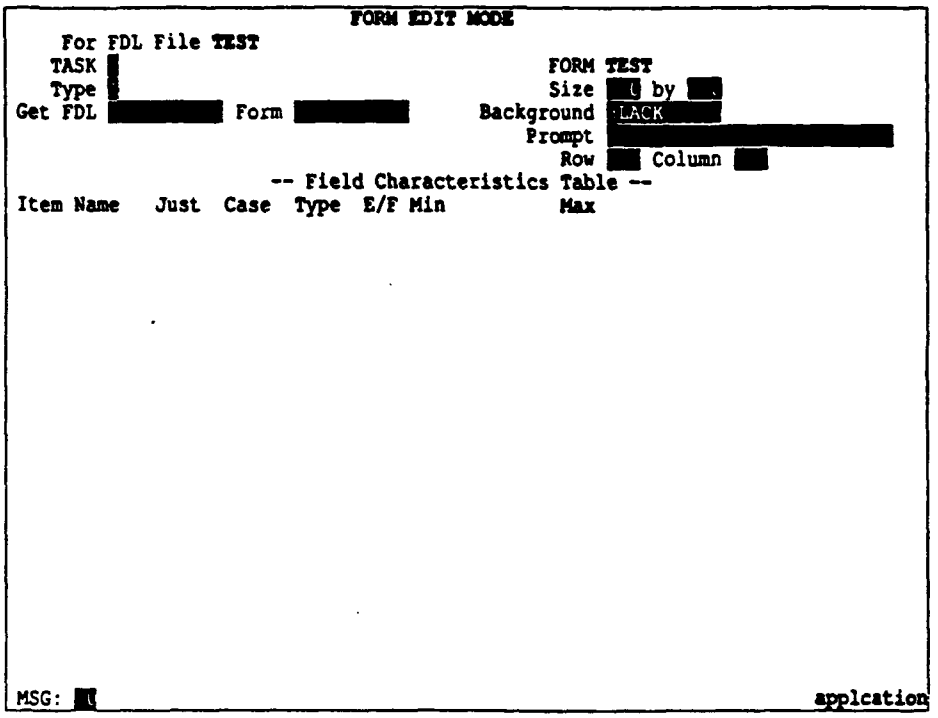


Figure A-11 Screen 11

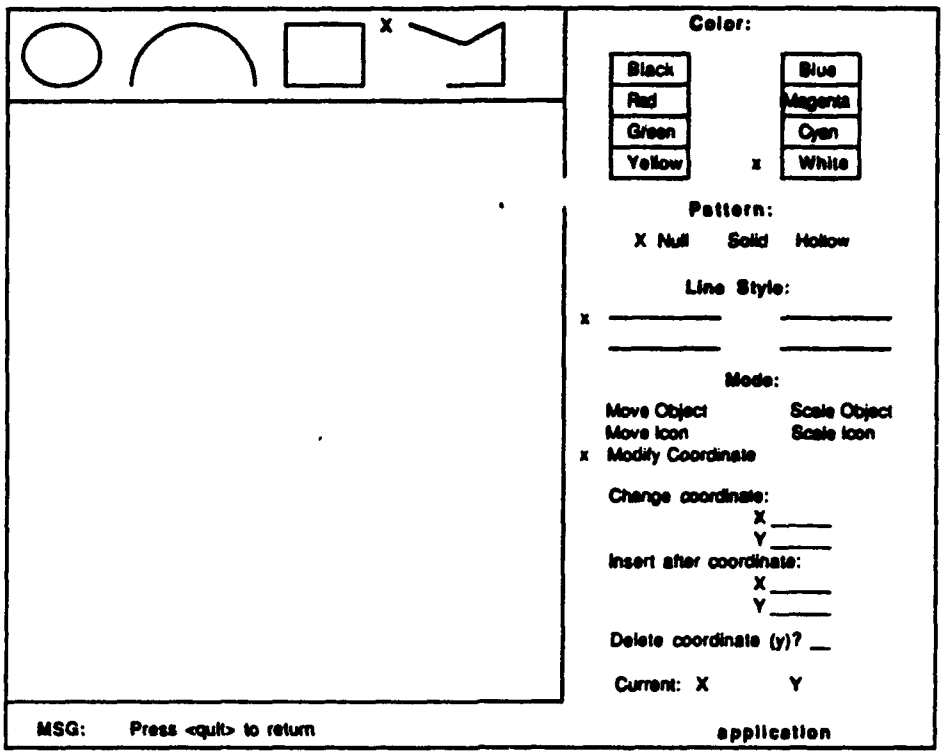
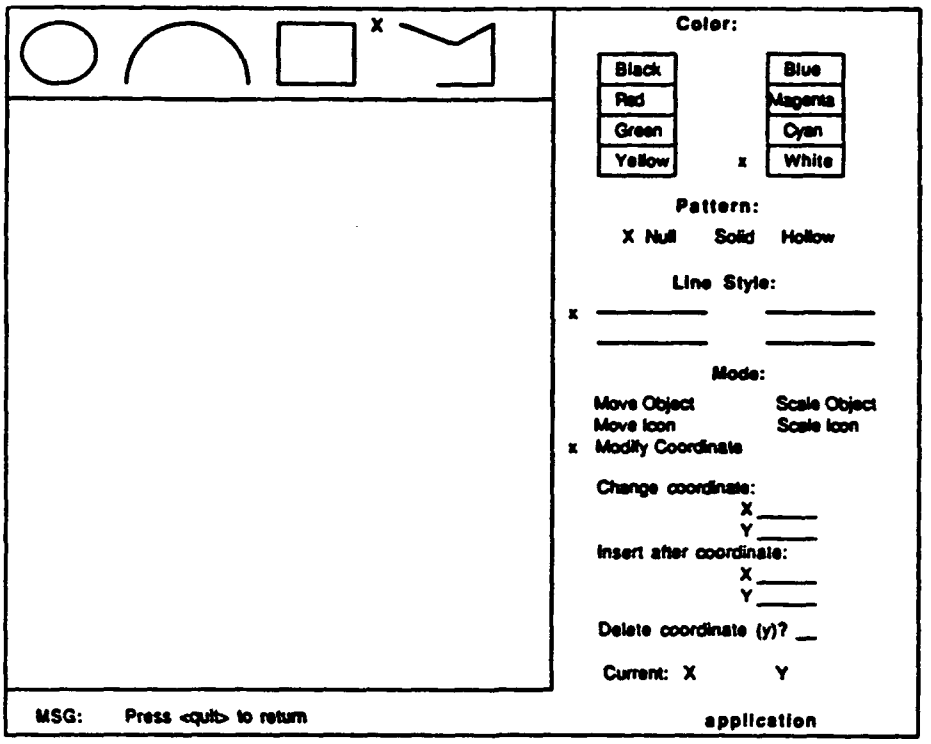
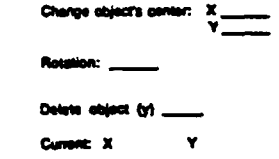
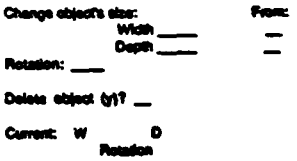


Figure A-12 Screen 12

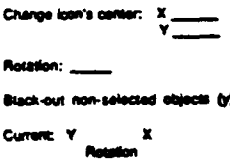




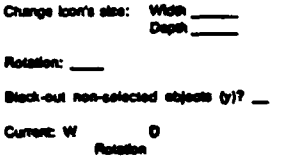
Move object Center Function



Change object size function



Move icon center Function



Change icon size function

Figure A-13 Screen 13

APPENDIX B

Icon Editor Mode User Interface

As part of the 2-D graphics support, icons are provided as a form with graphical representation of some action. This would allow the user to pick an action to be performed (by picking the form) rather than a complicated sequence of commands. Icons are created using a FDL source language file and FLAN. The purpose of the Icon Editor mode is to provide the user with an improved user interface for the creation of icons. Using the Icon Editor will reduce the amount of time needed to create an icon using FLAN and a FDL source file, as the Icon Editor produces many lines of code for relativly simple creations of objects. Additionally, the user receives an immediate feedback of the appearance of the icon which aids in development.

The Icon Edit Mode is accessed by either an Edit Task Screen command line instruction or by choosing the icon mode of editing on the Edit Task Screen. The Edit Task Screen command line access is accomplished by entering an "mf 'form name' i" if Modifying an existing Form or "if 'form name' i" if Inserting a new Form. The screen selection is done by entering an 'i' in the appropriate row on the 'pic' column of the EDIT TASK SCREEN. The user will then be presented with the ICON EDIT SCREEN (see Figure B-1.)

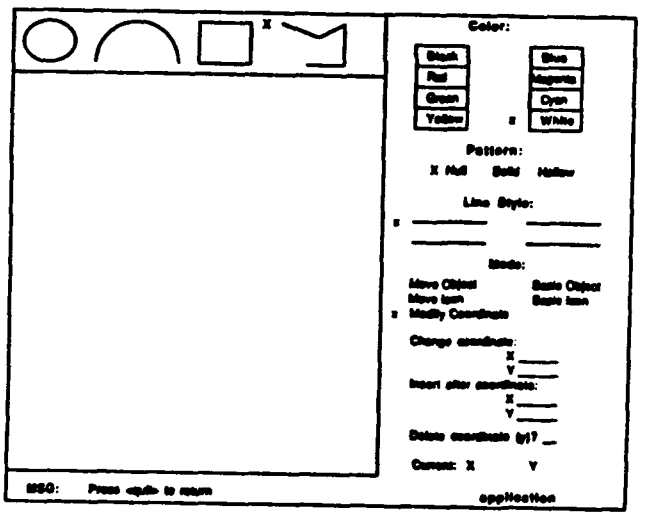
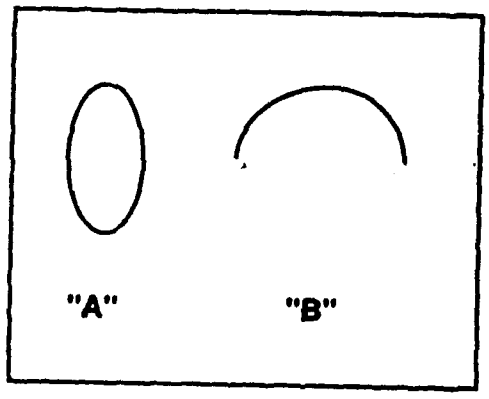


Figure B-1 Icon Editor opening screen

Creation of an object:

Icons are constructed using multiple objects. To create an object the user may select from four basic styles; circle, arc, box, and polyline. The circle may be stretched to form an ellipse, and then rotated about its center. The arc primitive (a partial circle) may also be stretched and rotated about its center. Note Figure B-2 demonstrates two such possiblities. The box can be stretched to make a rectangle in a similiar

fashion. The polyline primitive allows the user to create a single line or a series of connected lines. Figure B-3 shows the box and polyline primitives.



A- Circle with width greater than depth and rotated about its center.

B- Arc with width equal to depth no rotation.

Figure B-2 Creation of circle and arc primitives

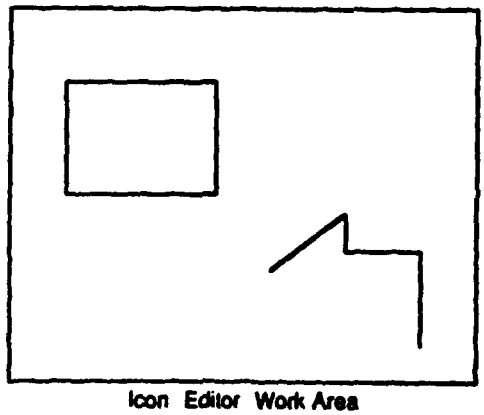


Figure B-3 Creation of box and polyline pimitives

To select one of the four styles, the user places the cursor on the icon which is to be created and presses <ENTER>.

To make an icon using the polyline primitive requires at least two coordinate points to fully describe a line. Three or more coordinates will make a series of connected lines.

See also Figure B-4 for an example:

o the cursor to be placed on the first coordinate with a <PF16> after this placement

o cursor placement for the second coordinate is selected similarly

o the last coordinate for the object is selected by positioning the cursor on the desired coordinate and then pressing <ENTER>

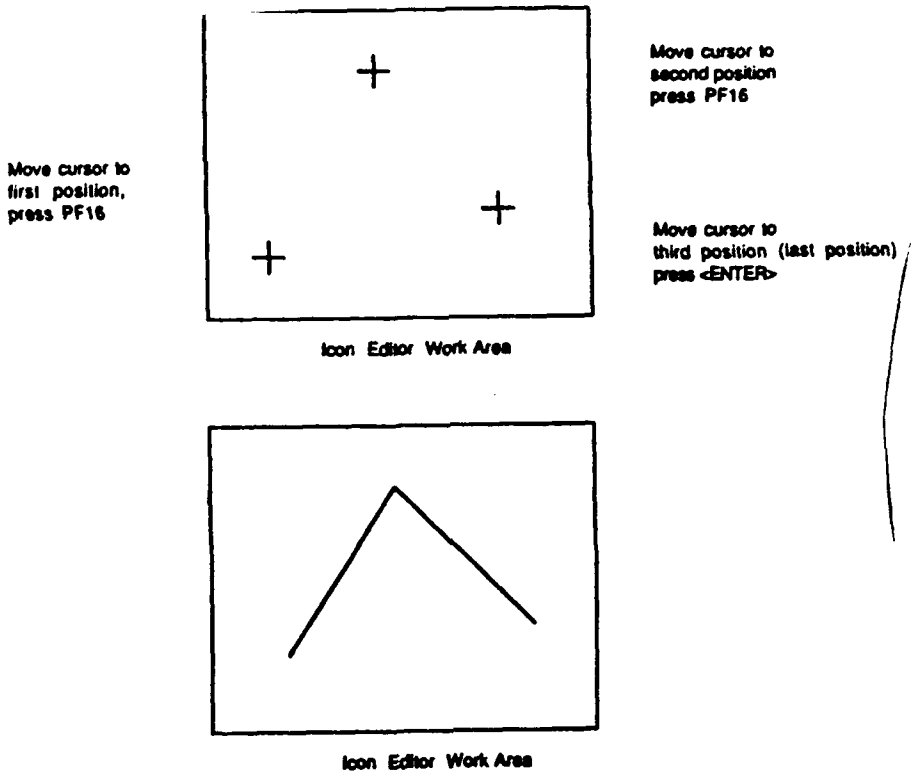


Figure B-4 Polyline primitive

An icon using the circle, arc, or box primitives is created by placing the cursor on the minimum x, y position before pressing <PF16>; similarly, the cursor is placed on the maximum x, y position of the object before pressing <ENTER>. See Figure B-5 for a representative example of the box formation.

Selection of an object:

An existing object is selected for modification or deletion by pressing <PF13> (for previous object) or <PF14> (for the next object) until the desired object is marked. A circle, arc, or box will have its width and depth marked with a dashed line. A polyline will be marked with a circle marker at each of its coordinates. The last coordinate of a polyline will be marked with a cross to show the default modification marker. Figures B-6a thru B-6c shows examples of the circle, box and polylines as they would appear when selected.

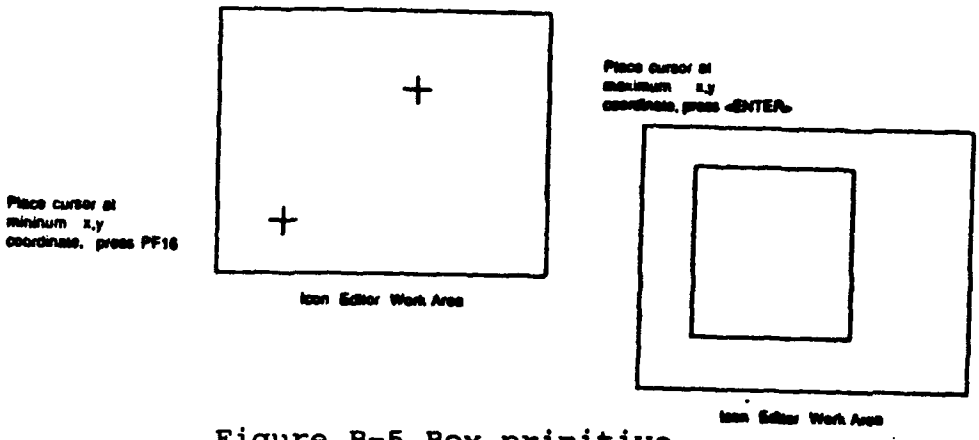


Figure B-5 Box primitive

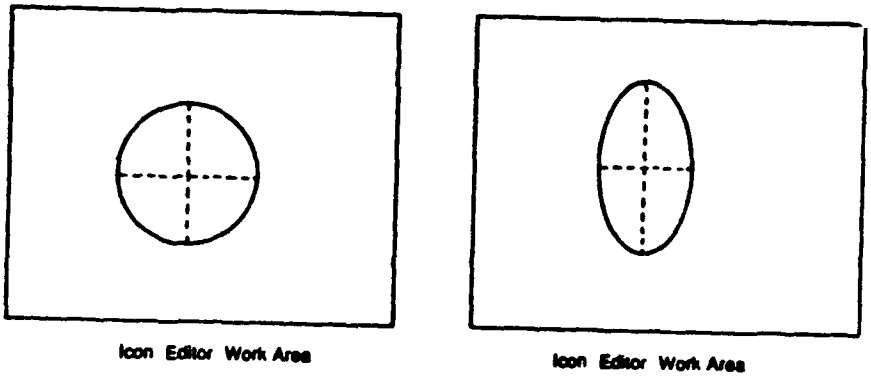


Figure B-6 (a) Selected circle

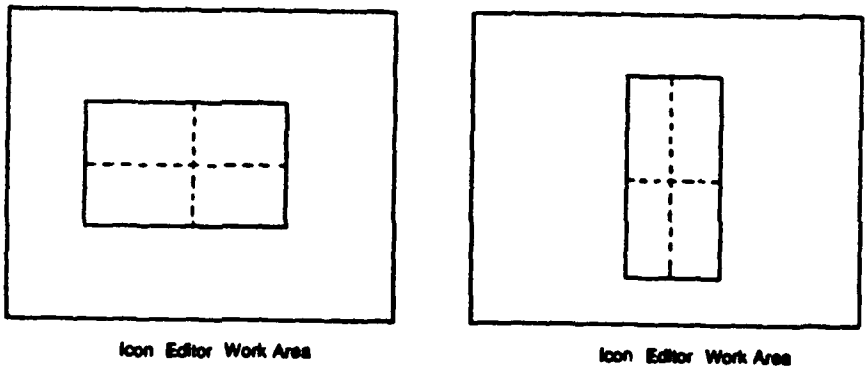


Figure B-6 (b) Selected box

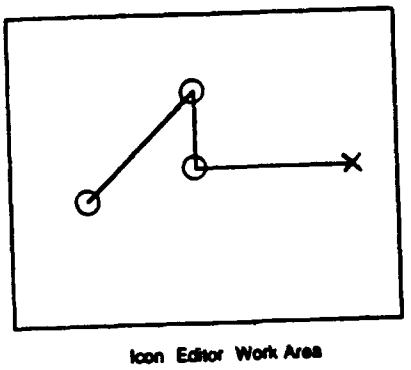


Figure B-6 (c) Selected polyline

Overlapping Objects:

Objects in an icon can appear to overlap (Figure B-7), or appear to be in front of, behind, or between other objects. Modifying such objects requires that the user imagine the objects are "STACKED" upon each other in some order. By selecting, and using the following commands, the user can place an object in the stack in a different position (behind, in front of, etc) other objects, as well as change its color, size, etc.

The user must first select an object, then the object can be put on the top of the stack, by pressing <PF15>. If the object is not explicitly put on the top of stack, it will be put back in its original position when another object is selected, or created, and when this editing mode is discontinued.

If an object is explicitly put on the top of the stack, it will not be returned to its original position, and it will remain in its current position on the stack.

If another object is selected after the currently selected object is explicitly placed on the top of stack, the currently selected object is unselected; it remains where it is on the stack and just below the object just selected.

If a new object is created after the currently selected object is explicitly put on the top of the stack,

- o the currently selected object is unselected
- o the old object remains where it is on the stack
- o and it is just below the object just created

For clarity, the user can blank-out all other objects in the stack, but the one selected. To blank-out the other objects, enter one of the modify icon modes and enter a 'Y' or 'y' to the prompt at the bottom of the change form (see Figure B-9).

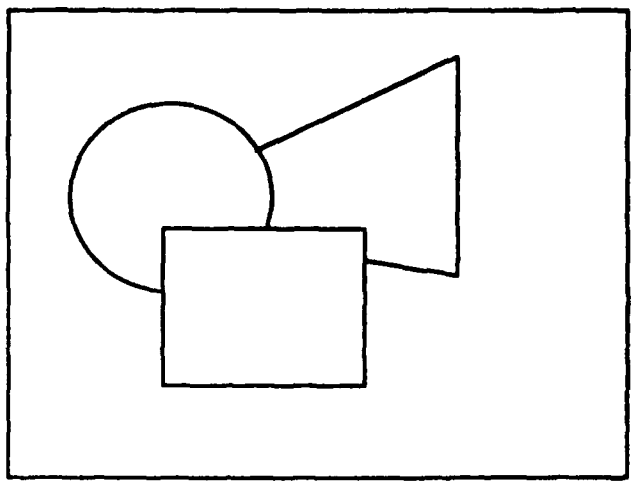


Figure B-7 Overlapping objects

Moving Objects:

Two functions are provided to change the position and size of the four basic styles of objects. The change function command area (see also Figure B-8) is located in the lower right corner of the screen.

To enter "move object's center function", the user tabs to move object and presses <ENTER>. When in this function, the screen resembles Figure B-8-(A). Once in this function, to move the center of the selected object:

- o to move up press <PF5>
- o to move down press <PF6>
- o to move left press <PF7>
- o to move right press <PF8>

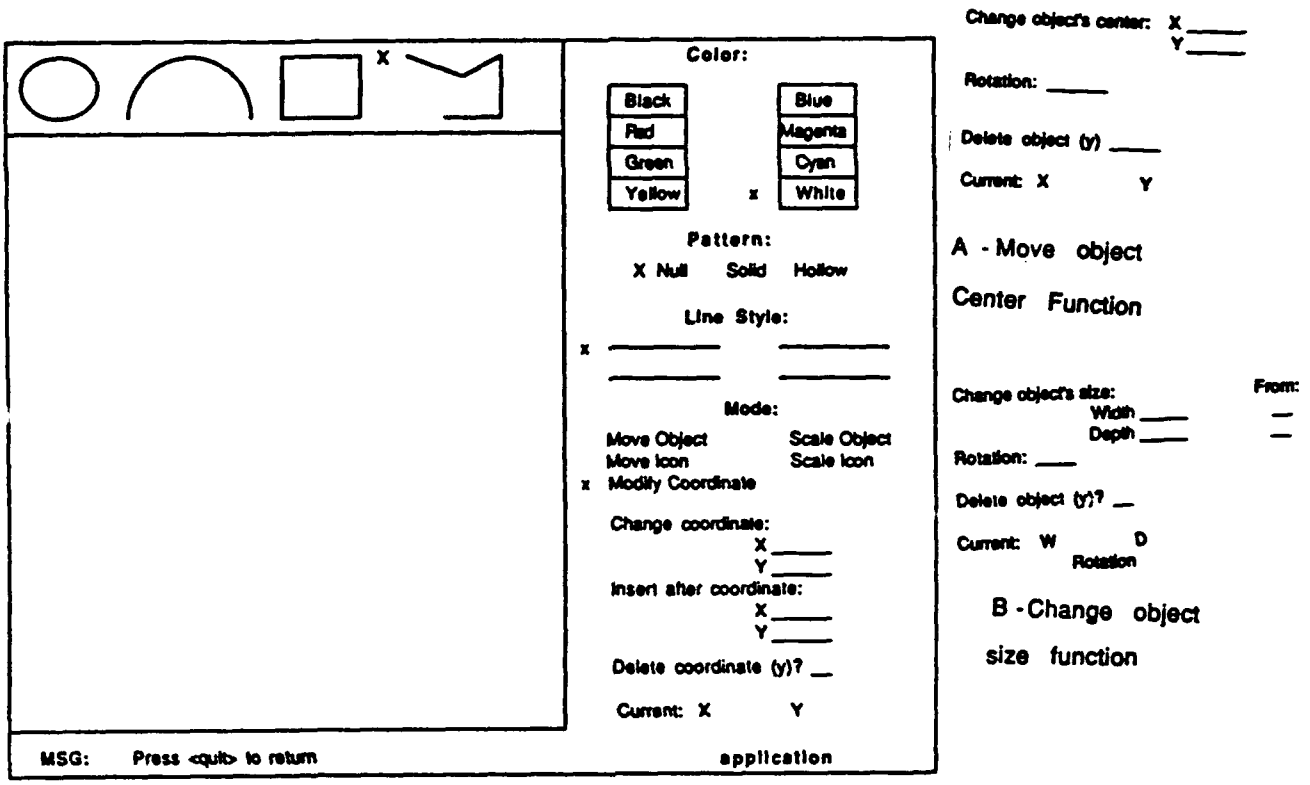


Figure B-8 Moving or changing an object
Showing default function (C)

Additionally to move the object, the new x,y of its center may be entered on the screen and the <ENTER> key pressed.

To enter "change object's dimensions function", the user tabs to scale object and presses <ENTER>. When in this function, the screen resembles Figure B-8-(B). In this function, to change the dimensions of the selected object:

- o to increase the depth, press <PF5>
- o to decrease its depth, press <PF6>
- o to increase its width, press <PF7>
- o to decrease its width, press <PF8>

The new dimensions can also be entered on the screen and the <ENTER> key pressed.

To enter "move polyline's coordinates function", the user tabs to the move coordinate function and presses <ENTER>. When in this function, the screen looks like Figure B-8-(C). In this function, to move the selected coordinate of a polyline:

- o to move it up, press <PF5>
- o to move it down, press <PF6>
- o to move it left, press <PF7>
- o to move it right, press <PF8>

The new x,y of coordinate of the polyline can also be entered on the screen and the <ENTER> key pressed.

To select a coordinate of a polyline selected for modification, the user repeatedly presses <PF9>, (for the previous coordinate) or, presses <PF10> (for the next coordinate), until the coordinate which is to be changed is marked as being selected.

Moving Icons:

Two functions are provided to change the position and size of an entire icon. The change function command area (see also Figure B-9) is located in the lower right corner of the screen.

To enter "move icon center function", the user tabs to move icon and presses <ENTER>. When in this function, the screen resembles Figure B-9-(A). Once in this function, to move the center of the selected icon:

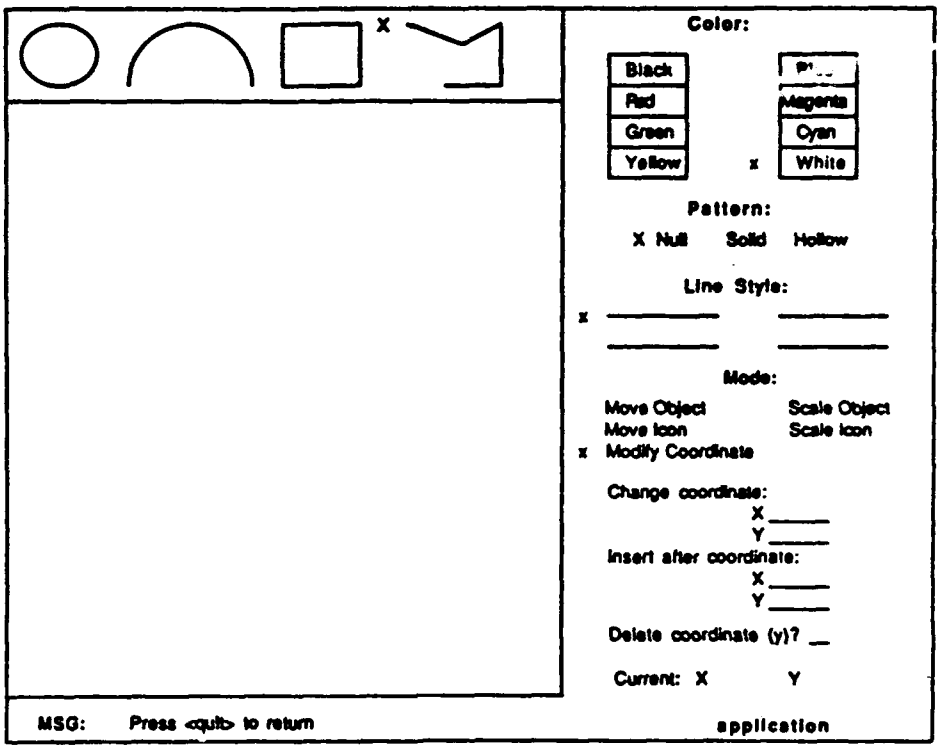
- o to move up press <PF5>
- o to move down press <PF6>
- o to move left press <PF7>
- o to move right press <PF8>

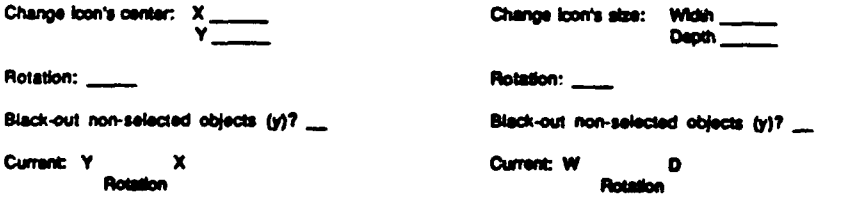
Additionally to move the icon, the new x,y of its center may be entered on the screen and the <ENTER> key pressed.

To enter "change icon dimensions function", the user tabs to scale icon and presses <ENTER>. When in this function, the screen resembles Figure B-9-(B). In this function, to change the dimensions of the selected icon:

- o to increase the depth, press <PF5>
- o to decrease its depth, press <PF6>
- o to increase its width, press <PF7>
- o to decrease its width, press <PF8>

The new dimensions can also be entered on the screen and the <ENTER> key pressed.





A - Move icon center Function B - Change Icon size function

Figure B-9 Moving or changing an icon

Scale objects from point other than center:

Objects may be scaled from a point other than the center by specifying a point on the object to "hold stationary". The allowed points to hold are: top, bottom, left side, right side. Within the change object size function screen (see Figure B-8-B) the user must specifying the hold-point after the width or depth (under From:). The valid entries are:

- o 'L' or 'l' after the new width if scaling from the left side
- o 'R' or 'r' after the new width if scaling from the right side
- o 'T' or 't' after the new depth if scaling from the top
- o 'B' or 'b' after the new depth if scaling from the bottom

Copy selected objects:

To copy the selected object, press <PF11>. The new object's center will be at the position which had been marked by the cursor when <PF11> was pressed.

Rotating:

When in a modify object mode;

The selected object can be rotated about its center. To rotate a selected object, the user presses <PF12> which will rotate the object counter-clockwize 10 degrees. Also, an exact positive or negative integer value for the rotation can be entered on the screen and the <ENTER> key pressed.

When in a modify Icon mode;

The icon can be rotated about its center. To rotate an, icon, the user presses <PF12> which will rotate the icon counter-clockwize 10 degrees. Also, an exact positive or negative integer value for the rotation can be entered on the screen and the <ENTER> key pressed.

Changing color:

To change the color or fill type used to draw objects the user tabs to the new color or fill pattern and presses <ENTER>. Changes must be made before object is drawn, to take effect. To change the color or fill pattern of an object, the user first selects the object. The selected object's color and fill become the current default. When these defaults are changed, the selected object's color and fill pattern is changed accordingly. The color white is the default.

Blank-out mode:

To blank-out all objects but the one selected, enter one of the modify object modes and enter a 'Y' or 'y' to the prompt.

Deleting an object:

To delete the currently selected object enter a 'y' into the field folowing 'Delete(y)?' and press <ENTER>.

Deleting a polyline coordinate:

To delete a coordinate enter modify coordinate mode and enter a 'Y' or 'y' after delete coordinate prompt.

Inserting a polyline coordinate:

To insert a coordinate enter modify coordinate mode and enter new coordinates into the modify coordinate form.

Changing line style:

To change the line style, the user tabs to the line style as represented in the primitive area, and presses <ENTER>. The selected style will be marked with an X. The Solid style is the default style.

Keypad Help:

Keypad Help is available by pressing <PF17>.

U.S. GOVERNMENT PRINTING OFFICE: 1992 - 648-127/62044